**PL/SQL Programming**

**Exercise 1: Control Structures**

**Code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE customers\_loans';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

**/**

CREATE TABLE customers\_loans (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

age NUMBER,

balance NUMBER,

isvip VARCHAR2(5),

interest\_rate NUMBER,

due\_date DATE

);

INSERT INTO customers\_loans VALUES (1, 'Tommy', 68, 7000, 'FALSE', 6.0, SYSDATE + 50);

INSERT INTO customers\_loans VALUES (2, 'Rohan', 45, 12000, 'FALSE', 5.5, SYSDATE + 60);

INSERT INTO customers\_loans VALUES (3, 'Sunny', 30, 9000, 'FALSE', 6.8, SYSDATE + 10);

INSERT INTO customers\_loans VALUES (4, 'Vicky', 70, 13000, 'FALSE', 5.0, SYSDATE + 40);

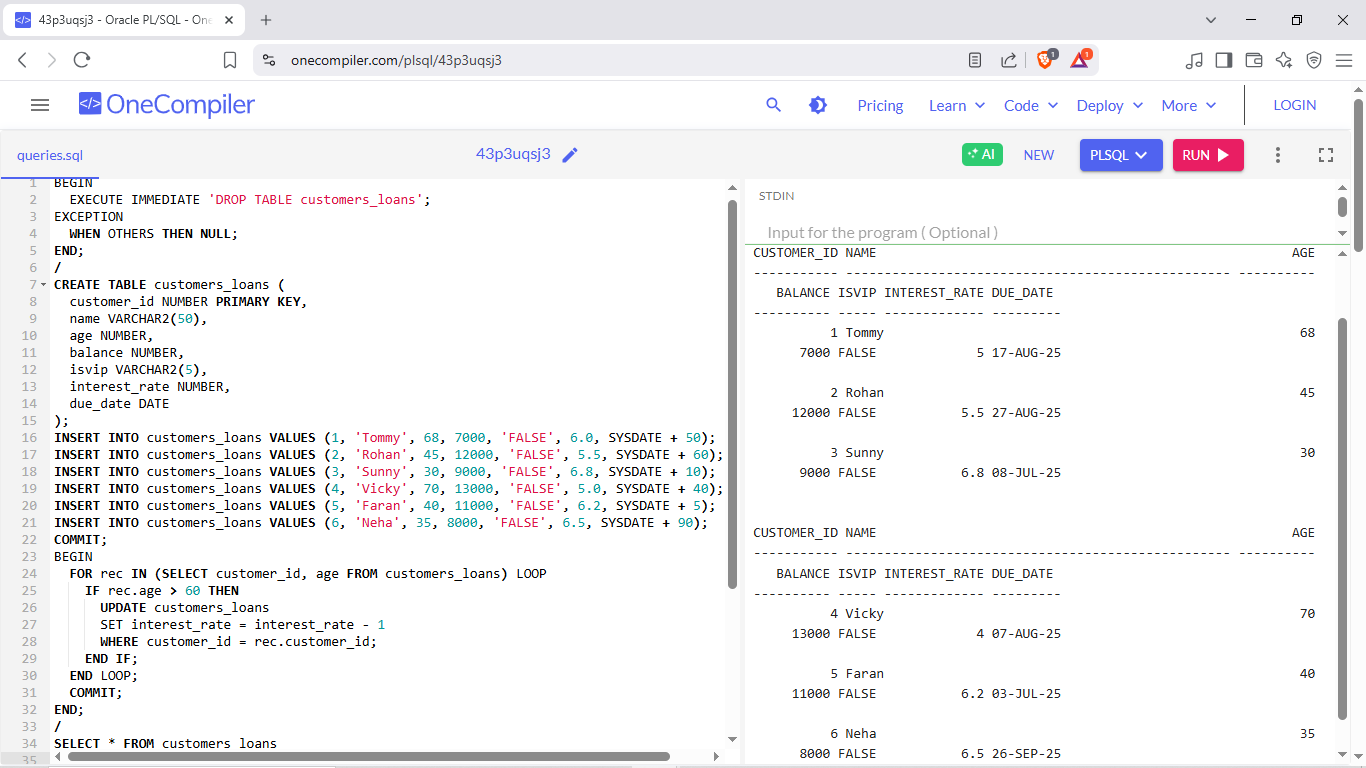
INSERT INTO customers\_loans VALUES (5, 'Faran', 40, 11000, 'FALSE', 6.2, SYSDATE + 5);

INSERT INTO customers\_loans VALUES (6, 'Neha', 35, 8000, 'FALSE', 6.5, SYSDATE + 90);

COMMIT;

select \* from customers\_loans;

--Table is Created



**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60

years old.

Write a PL/SQL block that loops through all customers, checks their age, and if

**Code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE customers\_loans';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

**/**

CREATE TABLE customers\_loans (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

age NUMBER,

balance NUMBER,

isvip VARCHAR2(5),

interest\_rate NUMBER,

due\_date DATE

);

INSERT INTO customers\_loans VALUES (1, 'Tommy', 68, 7000, 'FALSE', 6.0, SYSDATE + 50);

INSERT INTO customers\_loans VALUES (2, 'Rohan', 45, 12000, 'FALSE', 5.5, SYSDATE + 60);

INSERT INTO customers\_loans VALUES (3, 'Sunny', 30, 9000, 'FALSE', 6.8, SYSDATE + 10);

INSERT INTO customers\_loans VALUES (4, 'Vicky', 70, 13000, 'FALSE', 5.0, SYSDATE + 40);

INSERT INTO customers\_loans VALUES (5, 'Faran', 40, 11000, 'FALSE', 6.2, SYSDATE + 5);

INSERT INTO customers\_loans VALUES (6, 'Neha', 35, 8000, 'FALSE', 6.5, SYSDATE + 90);

COMMIT;

BEGIN

FOR rec IN (SELECT customer\_id, age FROM customers\_loans) LOOP

IF rec.age > 60 THEN

UPDATE customers\_loans

SET interest\_rate = interest\_rate - 1

WHERE customer\_id = rec.customer\_id;

END IF;

END LOOP;

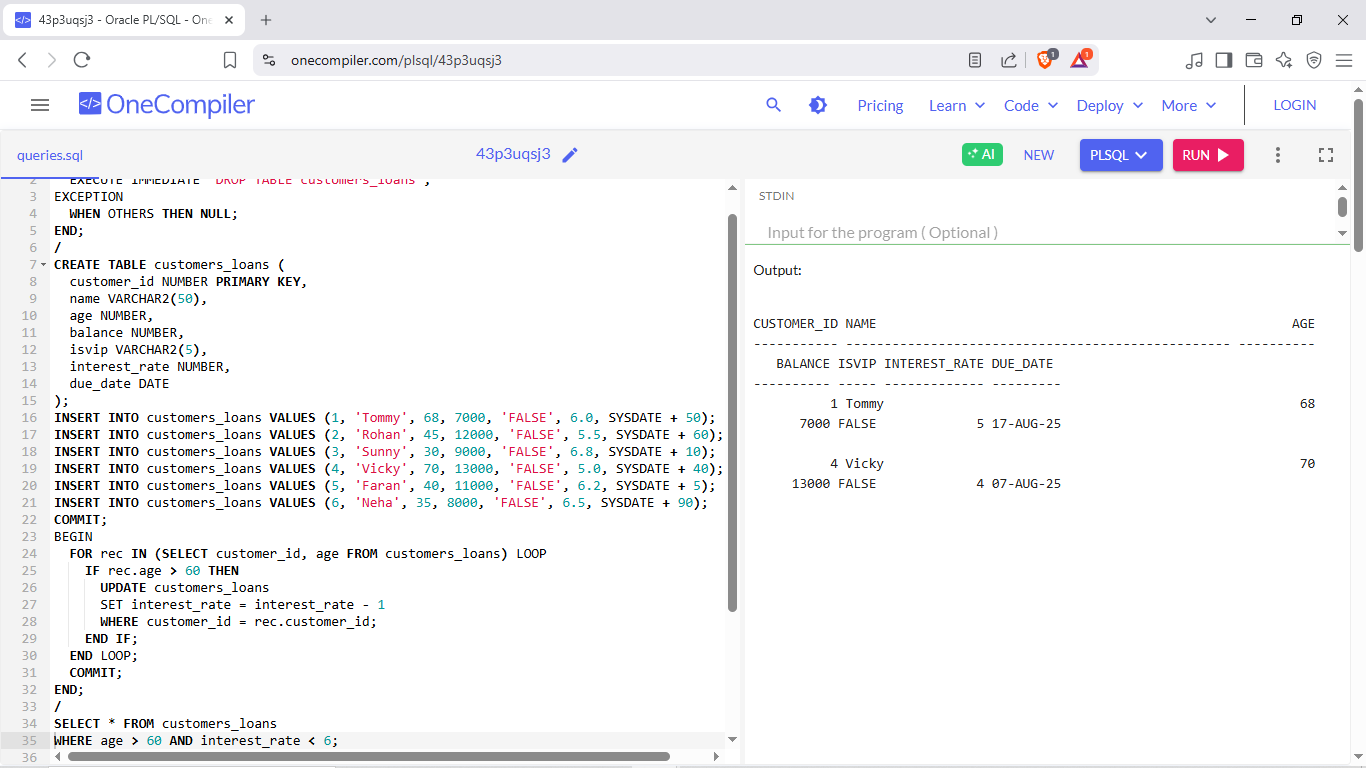
COMMIT;

END;

**/**

SELECT \* FROM customers\_loans

WHERE age > 60 AND interest\_rate < 6;

**Output:**

**Scenario 2:** A customer can be promoted to VIP status based on their balance.

Write a PL/SQL block that iterates through all customers and sets a flag IsVIP

to TRUE for those with a balance over $10,000.

**Code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE customers\_loans';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE customers\_loans (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

age NUMBER,

balance NUMBER,

isvip VARCHAR2(5),

interest\_rate NUMBER,

due\_date DATE

);

INSERT INTO customers\_loans VALUES (1, 'Tommy', 68, 7000, 'FALSE', 6.0, SYSDATE + 50);

INSERT INTO customers\_loans VALUES (2, 'Rohan', 45, 12000, 'FALSE', 5.5, SYSDATE + 60);

INSERT INTO customers\_loans VALUES (3, 'Sunny', 30, 9000, 'FALSE', 6.8, SYSDATE + 10);

INSERT INTO customers\_loans VALUES (4, 'Vicky', 70, 13000, 'FALSE', 5.0, SYSDATE + 40);

INSERT INTO customers\_loans VALUES (5, 'Faran', 40, 11000, 'FALSE', 6.2, SYSDATE + 5);

INSERT INTO customers\_loans VALUES (6, 'Neha', 35, 8000, 'FALSE', 6.5, SYSDATE + 90);

COMMIT;

BEGIN

FOR rec IN (SELECT customer\_id, balance FROM customers\_loans) LOOP

IF rec.balance > 10000 THEN

UPDATE customers\_loans SET isvip = 'TRUE'

WHERE customer\_id = rec.customer\_id;

END IF;

END LOOP;

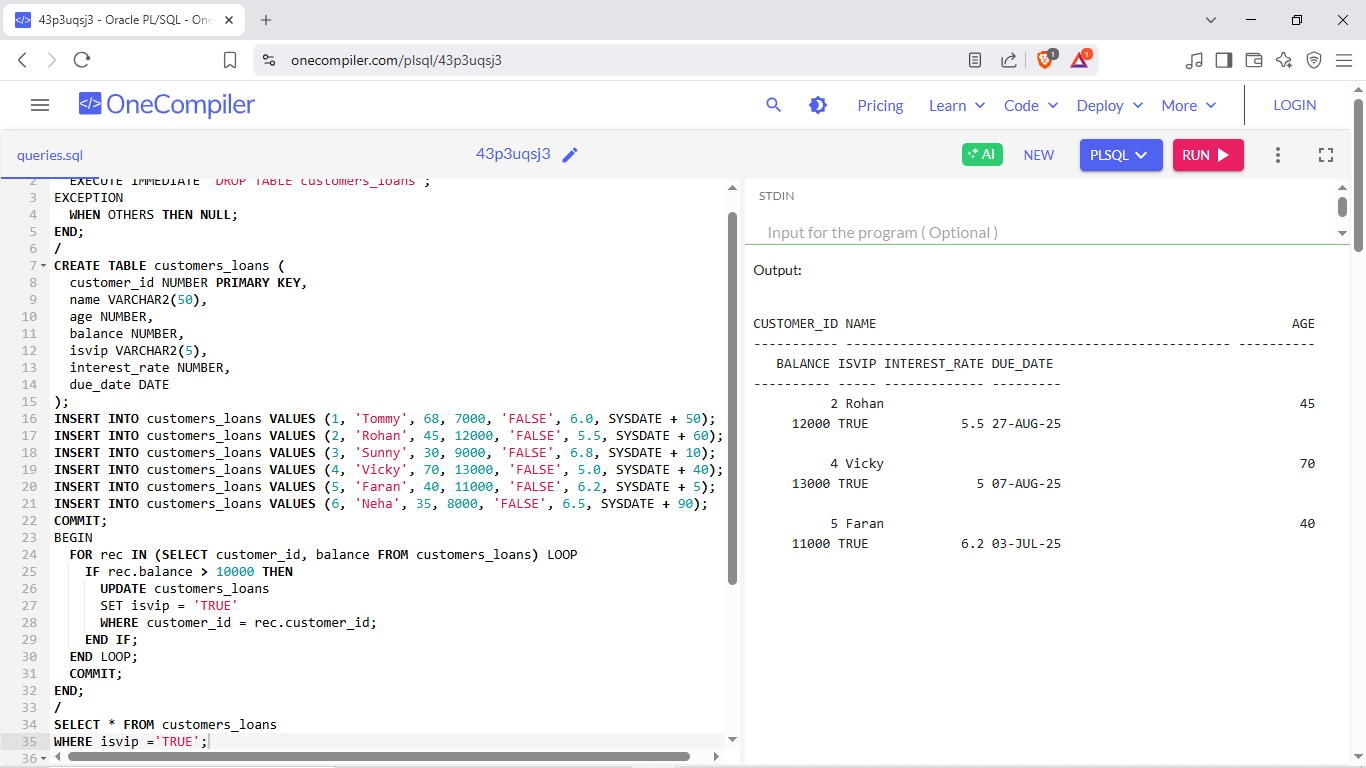
COMMIT;

END;

**/**

SELECT \* FROM customers\_loans

WHERE isvip ='TRUE';

**Output:**

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the

next 30 days. Write a PL/SQL block that fetches all loans due in the next 30 days and prints a

reminder message for each customer.

**Code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE customers\_loans';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

CREATE TABLE customers\_loans (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

age NUMBER,

balance NUMBER,

isvip VARCHAR2(5),

interest\_rate NUMBER,

due\_date DATE

);

INSERT INTO customers\_loans VALUES (1, 'Tommy', 68, 7000, 'FALSE', 6.0, SYSDATE + 50);

INSERT INTO customers\_loans VALUES (2, 'Rohan', 45, 12000, 'FALSE', 5.5, SYSDATE + 60);

INSERT INTO customers\_loans VALUES (3, 'Sunny', 30, 9000, 'FALSE', 6.8, SYSDATE + 10);

INSERT INTO customers\_loans VALUES (4, 'Vicky', 70, 13000, 'FALSE', 5.0, SYSDATE + 40);

INSERT INTO customers\_loans VALUES (5, 'Faran', 40, 11000, 'FALSE', 6.2, SYSDATE + 5);

INSERT INTO customers\_loans VALUES (6, 'Neha', 35, 8000, 'FALSE', 6.5, SYSDATE + 90);

COMMIT;

BEGIN

FOR rec IN (SELECT name, due\_date

FROM customers\_loans

WHERE due\_date BETWEEN SYSDATE AND SYSDATE + 30

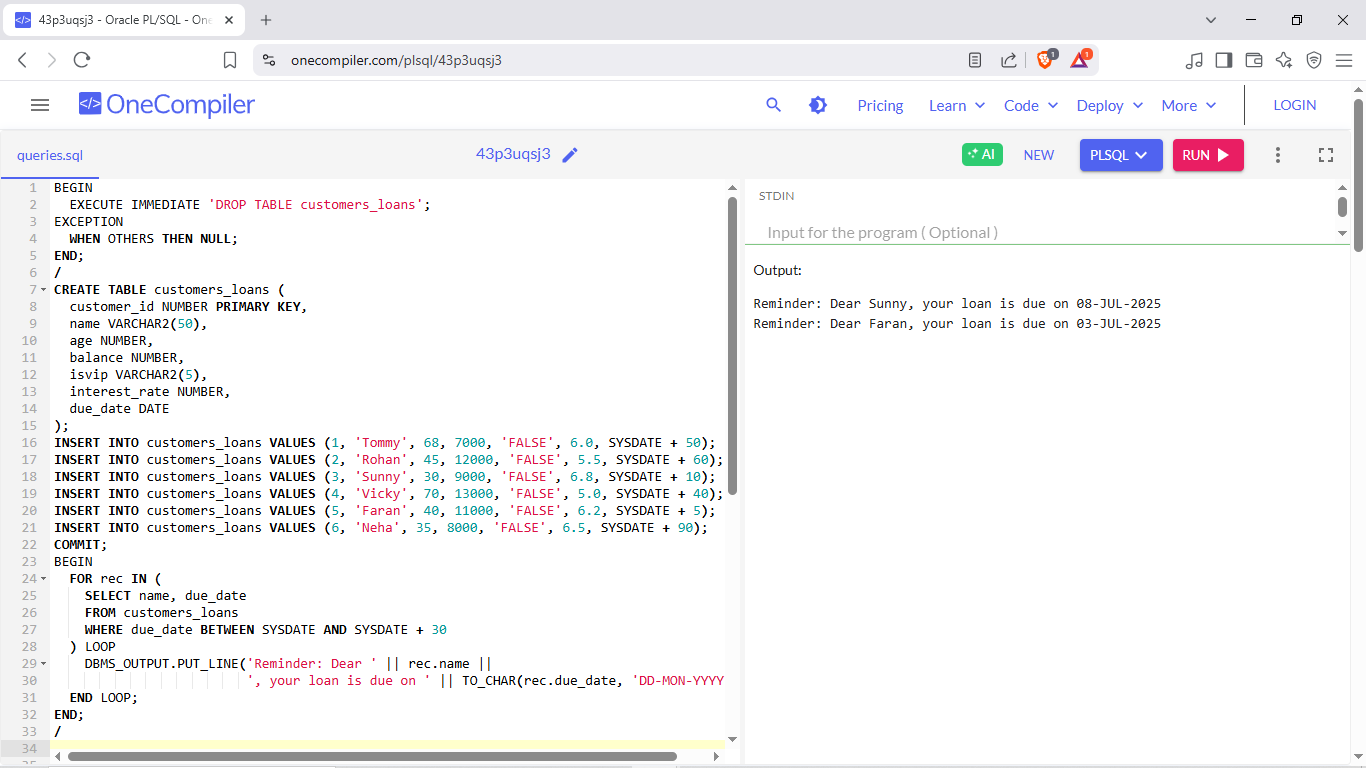
) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Dear ' || rec.name || ', your loan is due on ' || TO\_CHAR(rec.due\_date, 'DD-MON-YYYY'));

END LOOP;

END;

**/**

**Output:**

**Exercise 3: Stored Procedures**

**Code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE accounts';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

**/**

CREATE TABLE accounts (

acc\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

acc\_type VARCHAR2(20),

department VARCHAR2(30),

balance NUMBER

);

INSERT INTO accounts VALUES (1, 'Amar', 'savings', NULL, 10000);

INSERT INTO accounts VALUES (2, 'Bala', 'savings', NULL, 20000);

INSERT INTO accounts VALUES (3, 'Seetha', 'savings', NULL, 5000);

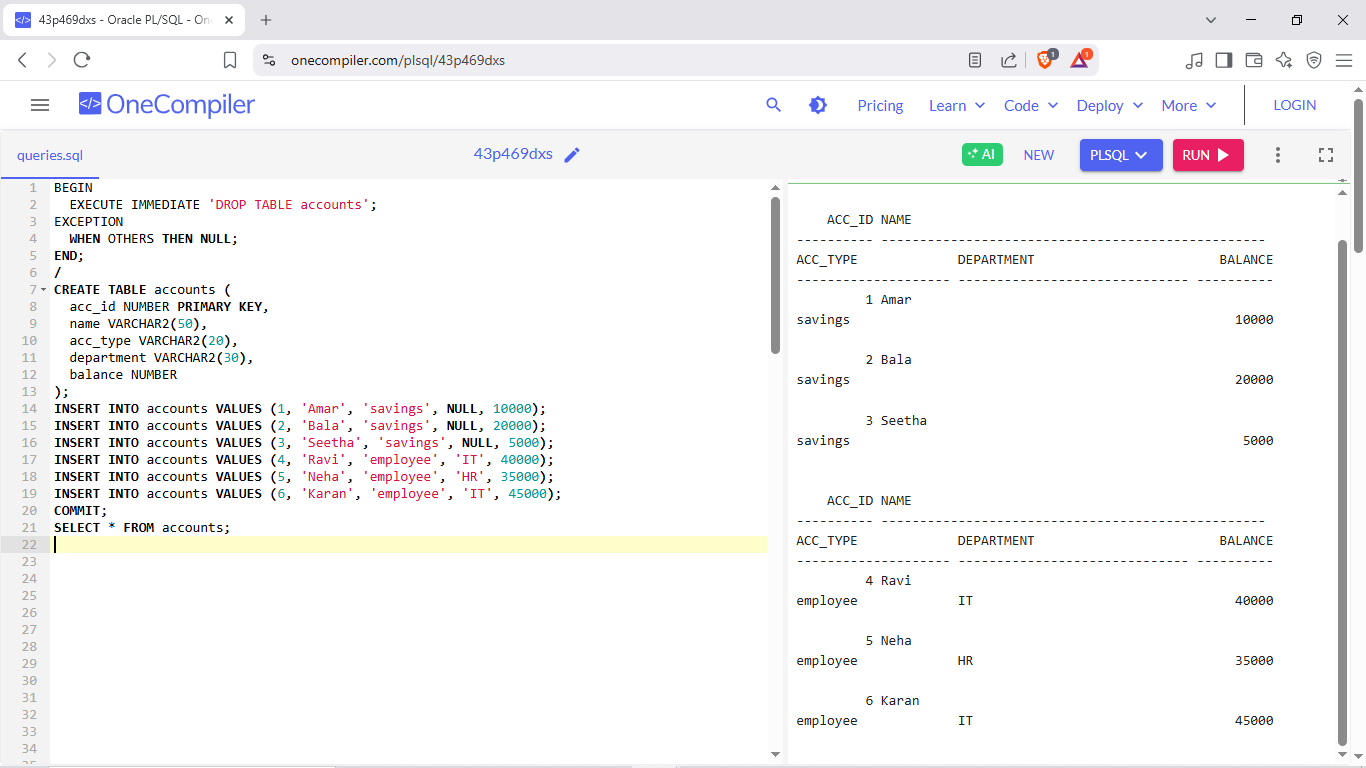
INSERT INTO accounts VALUES (4, 'Ravi', 'employee', 'IT', 40000);

INSERT INTO accounts VALUES (5, 'Neha', 'employee', 'HR', 35000);

INSERT INTO accounts VALUES (6, 'Karan', 'employee', 'IT', 45000);

COMMIT;

Table is Created



**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

Write a stored procedure ProcessMonthlyInterest that calculates and updates

the balance of all savings accounts by applying an interest rate of 1% to the current

Balance.

**Code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE accounts';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

**/**

CREATE TABLE accounts (

acc\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

acc\_type VARCHAR2(20),

department VARCHAR2(30),

balance NUMBER

);

INSERT INTO accounts VALUES (1, 'Amar', 'savings', NULL, 10000);

INSERT INTO accounts VALUES (2, 'Bala', 'savings', NULL, 20000);

INSERT INTO accounts VALUES (3, 'Seetha', 'savings', NULL, 5000);

INSERT INTO accounts VALUES (4, 'Ravi', 'employee', 'IT', 40000);

INSERT INTO accounts VALUES (5, 'Neha', 'employee', 'HR', 35000);

INSERT INTO accounts VALUES (6, 'Karan', 'employee', 'IT', 45000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

UPDATE savings\_accounts

SET balance = balance + (balance \* 0.01);

END;

**/**

BEGIN

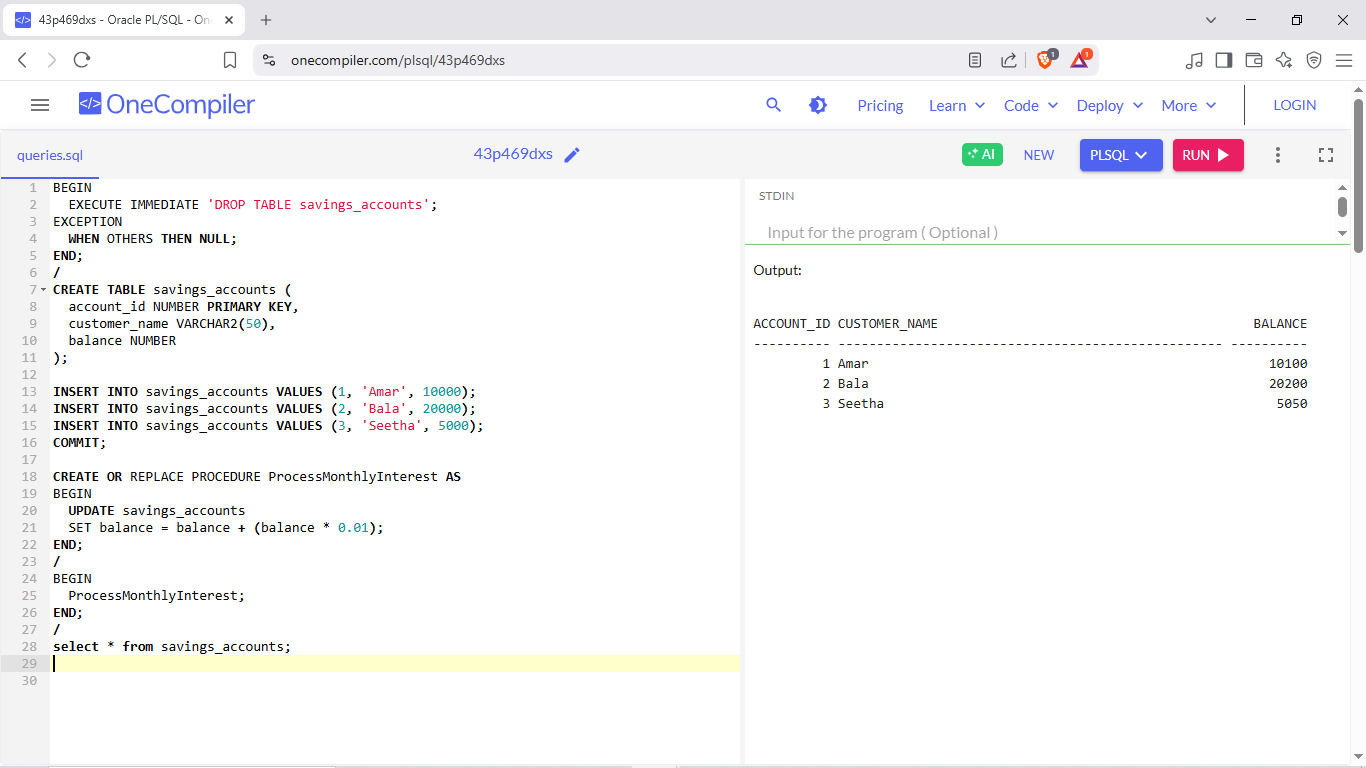
ProcessMonthlyInterest;

END;

**/**

select \* from savings\_accounts;

**Output:**



**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their

performance.

Write a stored procedure UpdateEmployeeBonus that updates the salary of

employees in a given department by adding a bonus percentage passed as a parameter**.**

**Code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE accounts';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

**/**

CREATE TABLE accounts (

acc\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

acc\_type VARCHAR2(20),

department VARCHAR2(30),

balance NUMBER

);

INSERT INTO accounts VALUES (1, 'Amar', 'savings', NULL, 10000);

INSERT INTO accounts VALUES (2, 'Bala', 'savings', NULL, 20000);

INSERT INTO accounts VALUES (3, 'Seetha', 'savings', NULL, 5000);

INSERT INTO accounts VALUES (4, 'Ravi', 'employee', 'IT', 40000);

INSERT INTO accounts VALUES (5, 'Neha', 'employee', 'HR', 35000);

INSERT INTO accounts VALUES (6, 'Karan', 'employee', 'IT', 45000);

COMMIT;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_name IN VARCHAR2,

bonus\_percent IN NUMBER

) AS

BEGIN

UPDATE accounts

SET balance = balance + (balance \* bonus\_percent / 100)

WHERE acc\_type = 'employee' and department=dept\_name;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to ' || dept\_name || ' department.');

END;

**/**

BEGIN

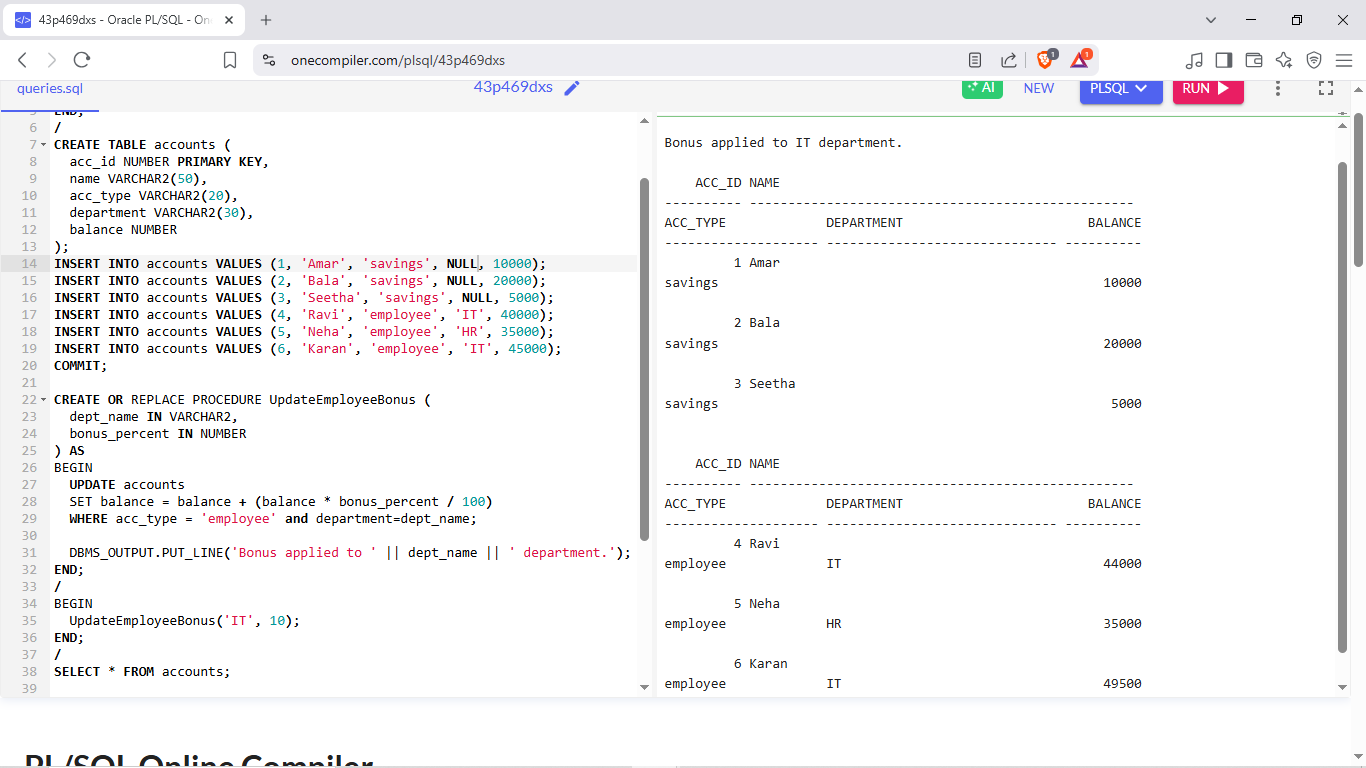
UpdateEmployeeBonus('IT', 10);

END;

**/**

SELECT \* FROM accounts;

**Output:**



**Scenario 3:** Customers should be able to transfer funds between their accounts.

Write a stored procedure TransferFunds that transfers a specified amount

from one account to another, checking that the source account has sufficient balance

before making the transfer.

**Code:**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE accounts';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

**/**

CREATE TABLE accounts (

acc\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

acc\_type VARCHAR2(20),

department VARCHAR2(30),

balance NUMBER

);

INSERT INTO accounts VALUES (1, 'Amar', 'savings', NULL, 10000);

INSERT INTO accounts VALUES (2, 'Bala', 'savings', NULL, 20000);

INSERT INTO accounts VALUES (3, 'Seetha', 'savings', NULL, 5000);

INSERT INTO accounts VALUES (4, 'Ravi', 'employee', 'IT', 40000);

INSERT INTO accounts VALUES (5, 'Neha', 'employee', 'HR', 35000);

INSERT INTO accounts VALUES (6, 'Karan', 'employee', 'IT', 45000);

COMMIT;

CREATE OR REPLACE PROCEDURE TransferFunds (

from\_id IN NUMBER,

to\_id IN NUMBER,

amount IN NUMBER

) AS

from\_balance NUMBER;

BEGIN

SELECT balance INTO from\_balance FROM accounts WHERE acc\_id = from\_id;

IF from\_balance < amount THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance in source account.'|| from\_id);

ELSE

UPDATE accounts SET balance = balance - amount

WHERE acc\_id = from\_id;

UPDATE accounts SET balance = balance + amount

WHERE acc\_id = to\_id;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful: ' || amount || ' transferred.');

END IF;

END;

**/**

BEGIN

TransferFunds(6, 5, 7000);

END;

**/**

SELECT \* FROM accounts;

**Output:**

